autogluon_each_location

October 7, 2023

[1]:
```
# config
run_analysis = False
```

[2]:
```
import pandas as pd
import numpy as np



import warnings
warnings.filterwarnings("ignore")

def fix_datetime(X, name):
    # Convert 'date_forecast' to datetime format and replace original column␣
 ↪with 'ds'
    X['ds'] = pd.to_datetime(X['date_forecast'])
    X.drop(columns=['date_forecast'], inplace=True, errors='ignore')
    X.sort_values(by='ds', inplace=True)
    X.set_index('ds', inplace=True)

    # Drop rows where the minute part of the time is not 0
    X = X[X.index.minute == 0]
    return X



def convert_to_datetime(X_train_observed, X_train_estimated, X_test, y_train):
    X_train_observed = fix_datetime(X_train_observed, "X_train_observed")
    X_train_estimated = fix_datetime(X_train_estimated, "X_train_estimated")
    X_test = fix_datetime(X_test, "X_test")

    # add sample weights, which are 1 for observed and 3 for estimated
    X_train_observed["sample_weight"] = 1
    X_train_estimated["sample_weight"] = 3
    X_test["sample_weight"] = 3

    X_train_observed["estimated_diff_hours"] = 0
```

```python
    X_train_estimated["estimated_diff_hours"] = (X_train_estimated.index - pd.to_datetime(X_train_estimated["date_calc"])).dt.total_seconds() / 3600
    X_test["estimated_diff_hours"] = (X_test.index - pd.to_datetime(X_test["date_calc"])).dt.total_seconds() / 3600

    X_train_estimated["estimated_diff_hours"] = X_train_estimated["estimated_diff_hours"].astype('int64')
    # the filled once will get dropped later anyways, when we drop y nans
    X_test["estimated_diff_hours"] = X_test["estimated_diff_hours"].fillna(-50).astype('int64')


    X_train_estimated.drop(columns=['date_calc'], inplace=True)
    X_test.drop(columns=['date_calc'], inplace=True)

    y_train['ds'] = pd.to_datetime(y_train['time'])
    y_train.drop(columns=['time'], inplace=True)
    y_train.sort_values(by='ds', inplace=True)
    y_train.set_index('ds', inplace=True)

    return X_train_observed, X_train_estimated, X_test, y_train




def preprocess_data(X_train_observed, X_train_estimated, X_test, y_train, location):
    # convert to datetime
    X_train_observed, X_train_estimated, X_test, y_train = convert_to_datetime(X_train_observed, X_train_estimated, X_test, y_train)

    y_train["y"] = y_train["pv_measurement"].astype('float64')
    y_train.drop(columns=['pv_measurement'], inplace=True)
    X_train = pd.concat([X_train_observed, X_train_estimated])

    # fill missng sample_weight with 3
    #X_train["sample_weight"] = X_train["sample_weight"].fillna(0)


    # clip all y values to 0 if negative
    y_train["y"] = y_train["y"].clip(lower=0)

    X_train = pd.merge(X_train, y_train, how="inner", left_index=True, right_index=True)

    # print number of nans in sample_weight
```

```python
    print(f"Number of nans in sample_weight: {X_train['sample_weight'].isna().
 ↪sum()}")
    # print number of nans in y
    print(f"Number of nans in y: {X_train['y'].isna().sum()}")


    X_train["location"] = location
    X_test["location"] = location

    return X_train, X_test
# Define locations
locations = ['A', 'B', 'C']

X_trains = []
X_tests = []
# Loop through locations
for loc in locations:
    print(f"Processing location {loc}...")
    # Read target training data
    y_train = pd.read_parquet(f'{loc}/train_targets.parquet')

    # Read estimated training data and add location feature
    X_train_estimated = pd.read_parquet(f'{loc}/X_train_estimated.parquet')

    # Read observed training data and add location feature
    X_train_observed= pd.read_parquet(f'{loc}/X_train_observed.parquet')

    # Read estimated test data and add location feature
    X_test_estimated = pd.read_parquet(f'{loc}/X_test_estimated.parquet')

    # Preprocess data
    X_train, X_test = preprocess_data(X_train_observed, X_train_estimated,␣
 ↪X_test_estimated, y_train, loc)

    X_trains.append(X_train)
    X_tests.append(X_test)

# Concatenate all data and save to csv
X_train = pd.concat(X_trains)
X_test = pd.concat(X_tests)
```

```
Processing location A…
Number of nans in sample_weight: 0
Number of nans in y: 0
Processing location B…
Number of nans in sample_weight: 0
Number of nans in y: 4
```

```
Processing location C…
Number of nans in sample_weight: 0
Number of nans in y: 6059
```

1 Feature enginering

[3]:
```
# temporary
X_train["hour"] = X_train.index.hour
X_train["weekday"] = X_train.index.weekday
# weekday or is_weekend
X_train["is_weekend"] = X_train["weekday"].apply(lambda x: 1 if x >= 5 else 0)

# drop weekday
#X_train.drop(columns=["weekday"], inplace=True)
X_train["month"] = X_train.index.month
X_train["year"] = X_train.index.year

X_test["hour"] = X_test.index.hour
X_test["weekday"] = X_test.index.weekday

# weekday or is_weekend
X_test["is_weekend"] = X_test["weekday"].apply(lambda x: 1 if x >= 5 else 0)

# drop weekday
#X_test.drop(columns=["weekday"], inplace=True)
X_test["month"] = X_test.index.month
X_test["year"] = X_test.index.year



to_drop = ["snow_drift:idx", "snow_density:kgm3"]

X_train.drop(columns=to_drop, inplace=True)
X_test.drop(columns=to_drop, inplace=True)

X_train.dropna(subset=['y'], inplace=True)
X_train.to_csv('X_train_raw.csv', index=True)
X_test.to_csv('X_test_raw.csv', index=True)
```

[4]:
```
import autogluon.eda.auto as auto

if run_analysis:
    auto.dataset_overview(train_data=X_train, test_data=X_test, label="y",␣
 ↪sample=None)
```

[5]:
```
if run_analysis:
    auto.target_analysis(train_data=X_train, label="y")
```

2 Starting

[6]:
```
import os


# Get the last submission number
last_submission_number = int(max([int(filename.split('_')[1].split('.')[0]) for␣
 ↪filename in os.listdir('submissions') if "submission" in filename]))
print("Last submission number:", last_submission_number)
print("Now creating submission number:", last_submission_number + 1)

# Create the new filename
new_filename = f'submission_{last_submission_number + 1}'

hello = os.environ.get('HELLO')
if hello is not None:
    new_filename += f'_{hello}'

print("New filename:", new_filename)
```

```
Last submission number: 78
Now creating submission number: 79
New filename: submission_79
```

[7]:
```
from autogluon.tabular import TabularDataset, TabularPredictor
train_data = TabularDataset('X_train_raw.csv')
train_data.drop(columns=['ds'], inplace=True)

label = 'y'
metric = 'mean_absolute_error'
time_limit = 60
presets = 'best_quality'

sample_weight = 'sample_weight' #None
weight_evaluation = True #False
```

[8]:
```
predictors = [None, None, None]
```

[9]:
```
loc = "A"
print(f"Training model for location {loc}...")
predictor = TabularPredictor(label=label, eval_metric=metric,␣
 ↪path=f"AutogluonModels/{new_filename}_{loc}", sample_weight=sample_weight,␣
 ↪weight_evaluation=weight_evaluation).fit(train_data[train_data["location"]␣
 ↪== loc], time_limit=time_limit, presets=presets)
predictors[0] = predictor
```

```
Warning: path already exists! This predictor may overwrite an existing
predictor! path="AutogluonModels/submission_79_A"
```

```
Presets specified: ['best_quality']
Stack configuration (auto_stack=True): num_stack_levels=1, num_bag_folds=8,
num_bag_sets=20
Values in column 'sample_weight' used as sample weights instead of predictive
features. Evaluation will report weighted metrics, so ensure same column exists
in test data.
Beginning AutoGluon training … Time limit = 60s
AutoGluon will save models to "AutogluonModels/submission_79_A/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.3.0: Mon Jan 30 20:38:37 PST 2023;
root:xnu-8792.81.3~2/RELEASE_ARM64_T6000
Disk Space Avail:   49.88 GB / 494.38 GB (10.1%)
Train Data Rows:    34061
Train Data Columns: 51
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (5733.42, 0.0, 631.01116,
1166.20607)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    3385.79 MB
        Train Data (Original)  Memory Usage: 15.33 MB (0.5% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 4 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 2): ['elevation:m', 'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
```

```
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 42 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  6 | ['estimated_diff_hours', 'hour', 'weekday',
'is_weekend', 'month', …]
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 39 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])       :  5 | ['estimated_diff_hours', 'hour',
'weekday', 'month', 'year']
                ('int', ['bool']) :  4 | ['is_day:idx', 'is_in_shadow:idx',
'wind_speed_w_1000hPa:ms', 'is_weekend']
        0.1s = Fit runtime
        48 features in original data used to generate 48 features in processed
data.
        Train Data (Processed) Memory Usage: 12.13 MB (0.4% of available memory)
Data preprocessing and feature engineering runtime = 0.16s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge'],
        'CAT': {},
        'XGB': {},
        'FASTAI': {},
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
AutoGluon will fit 2 stack levels (L1 to L2) …
```

```
Fitting 11 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 39.88s of the
59.84s of remaining time.

Training model for location A…

        -277.2896        = Validation score   (-mean_absolute_error)
        0.02s    = Training   runtime
        0.85s    = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 38.92s of the
58.87s of remaining time.
        -278.2945        = Validation score   (-mean_absolute_error)
        0.03s    = Training   runtime
        0.79s    = Validation runtime
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 38.05s of the
58.01s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -144.6382        = Validation score   (-mean_absolute_error)
        31.83s   = Training   runtime
        56.67s   = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 59.84s of the
16.76s of remaining time.
        -144.6382        = Validation score   (-mean_absolute_error)
        0.16s    = Training   runtime
        0.0s     = Validation runtime
Fitting 9 L2 models …
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 16.59s of the
16.59s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -145.9616        = Validation score   (-mean_absolute_error)
        2.66s    = Training   runtime
        0.57s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 11.65s of the 11.64s
of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -143.7107        = Validation score   (-mean_absolute_error)
        1.41s    = Training   runtime
        0.21s    = Validation runtime
Fitting model: RandomForestMSE_BAG_L2 … Training model for up to 8.51s of the
8.5s of remaining time.
        -143.636         = Validation score   (-mean_absolute_error)
        31.75s   = Training   runtime
        0.83s    = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L3 … Training model for up to 59.84s of the
```

```
-24.68s of remaining time.
        -142.1966        = Validation score   (-mean_absolute_error)
        0.18s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 84.89s … Best model:
"WeightedEnsemble_L3"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_79_A/")
```

[ ]:
```
loc = "B"
print(f"Training model for location {loc}...")
predictor = TabularPredictor(label=label, eval_metric=metric,␣
 ↪path=f"AutogluonModels/{new_filename}_{loc}", sample_weight=sample_weight,␣
 ↪weight_evaluation=weight_evaluation).fit(train_data[train_data["location"]␣
 ↪== loc], time_limit=time_limit, presets=presets)
predictors[1] = predictor
```

```
Warning: path already exists! This predictor may overwrite an existing
predictor! path="AutogluonModels/submission_79_jorge_B"
Presets specified: ['best_quality']
Stack configuration (auto_stack=True): num_stack_levels=1, num_bag_folds=8,
num_bag_sets=20
Values in column 'sample_weight' used as sample weights instead of predictive
features. Evaluation will report weighted metrics, so ensure same column exists
in test data.
Beginning AutoGluon training … Time limit = 60s
AutoGluon will save models to "AutogluonModels/submission_79_jorge_B/"
AutoGluon Version:  0.8.1
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   15.97 GB / 494.38 GB (3.2%)
Train Data Rows:    32819
Train Data Columns: 51
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (1152.3, -0.0, 96.89334, 194.00409)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    4394.88 MB
        Train Data (Original)  Memory Usage: 14.77 MB (0.3% of available memory)
```

```
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 4 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 2): ['elevation:m', 'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 42 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  6 | ['estimated_diff_hours', 'hour', 'weekday',
'is_weekend', 'month', …]
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 39 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])       :  5 | ['estimated_diff_hours', 'hour',
'weekday', 'month', 'year']
                ('int', ['bool']) :  4 | ['is_day:idx', 'is_in_shadow:idx',
'wind_speed_w_1000hPa:ms', 'is_weekend']
        0.1s = Fit runtime
        48 features in original data used to generate 48 features in processed
data.
        Train Data (Processed) Memory Usage: 11.68 MB (0.3% of available memory)
Data preprocessing and feature engineering runtime = 0.16s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
```

```
'GBMLarge'],
        'CAT': {},
        'XGB': {},
        'FASTAI': {},
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
AutoGluon will fit 2 stack levels (L1 to L2) …
Fitting 11 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 39.88s of the
59.84s of remaining time.
```

```
Training model for location B…
```

```
        Not enough time to generate out-of-fold predictions for model. Estimated
time required was 160.16s compared to 51.82s of available time.
        Time limit exceeded… Skipping KNeighborsUnif_BAG_L1.
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 37.4s of the
57.35s of remaining time.
        Not enough time to generate out-of-fold predictions for model. Estimated
time required was 133.14s compared to 48.59s of available time.
        Time limit exceeded… Skipping KNeighborsDist_BAG_L1.
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 35.32s of the
55.28s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -23.1587         = Validation score   (-mean_absolute_error)
        24.39s   = Training   runtime
        65.15s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 0.11s of the 20.06s
of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -116.8768        = Validation score   (-mean_absolute_error)
        1.21s    = Training   runtime
        0.02s    = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 59.84s of the
16.45s of remaining time.
```

```
        -23.1587         = Validation score   (-mean_absolute_error)
        0.12s    = Training   runtime
        0.0s     = Validation runtime
Fitting 9 L2 models …
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 16.32s of the
16.3s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -21.9002         = Validation score   (-mean_absolute_error)
        3.87s    = Training   runtime
        1.0s     = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 10.01s of the 10.0s
of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -21.3768         = Validation score   (-mean_absolute_error)
        1.57s    = Training   runtime
        0.18s    = Validation runtime
Fitting model: RandomForestMSE_BAG_L2 … Training model for up to 6.63s of the
6.62s of remaining time.
        -20.3273         = Validation score   (-mean_absolute_error)
        25.22s   = Training   runtime
        0.86s    = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L3 … Training model for up to 59.84s of the
-19.69s of remaining time.
        -20.3273         = Validation score   (-mean_absolute_error)
        0.2s     = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 79.91s … Best model:
"WeightedEnsemble_L3"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_79_jorge_B/")
```

[ ]:
```
loc = "C"
print(f"Training model for location {loc}...")
predictor = TabularPredictor(label=label, eval_metric=metric,␣
  ↪path=f"AutogluonModels/{new_filename}_{loc}", sample_weight=sample_weight,␣
  ↪weight_evaluation=weight_evaluation).fit(train_data[train_data["location"]␣
  ↪== loc], time_limit=time_limit, presets=presets)
predictors[2] = predictor
```

```
Warning: path already exists! This predictor may overwrite an existing
predictor! path="AutogluonModels/submission_79_jorge_C"
Presets specified: ['best_quality']
Stack configuration (auto_stack=True): num_stack_levels=1, num_bag_folds=8,
num_bag_sets=20
Values in column 'sample_weight' used as sample weights instead of predictive
```

```
features. Evaluation will report weighted metrics, so ensure same column exists
in test data.
Beginning AutoGluon training … Time limit = 60s
AutoGluon will save models to "AutogluonModels/submission_79_jorge_C/"
AutoGluon Version:  0.8.1
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   15.55 GB / 494.38 GB (3.1%)
Train Data Rows:    26071
Train Data Columns: 51
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and label-values can't be converted to int).
        Label info (max, min, mean, stddev): (999.6, -0.0, 77.70004, 165.87752)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    4399.47 MB
        Train Data (Original)  Memory Usage: 11.73 MB (0.3% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 3 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 2): ['elevation:m', 'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 42 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
```

```
'clear_sky_rad:W', …]
                ('int', [])   :  6 | ['estimated_diff_hours', 'hour', 'weekday',
'is_weekend', 'month', …]
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])       :  5 | ['estimated_diff_hours', 'hour',
'weekday', 'month', 'year']
                ('int', ['bool']) :  3 | ['is_day:idx', 'is_in_shadow:idx',
'is_weekend']
        0.1s = Fit runtime
        48 features in original data used to generate 48 features in processed
data.
        Train Data (Processed) Memory Usage: 9.46 MB (0.2% of available memory)
Data preprocessing and feature engineering runtime = 0.14s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}, {},
'GBMLarge'],
        'CAT': {},
        'XGB': {},
        'FASTAI': {},
        'RF': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'XT': [{'criterion': 'gini', 'ag_args': {'name_suffix': 'Gini',
'problem_types': ['binary', 'multiclass']}}, {'criterion': 'entropy', 'ag_args':
{'name_suffix': 'Entr', 'problem_types': ['binary', 'multiclass']}},
{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
AutoGluon will fit 2 stack levels (L1 to L2) …
Fitting 11 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 … Training model for up to 39.9s of the
59.86s of remaining time.

Training model for location C…
```

```
        Not enough time to generate out-of-fold predictions for model. Estimated
time required was 105.14s compared to 51.84s of available time.
        Time limit exceeded… Skipping KNeighborsUnif_BAG_L1.
Fitting model: KNeighborsDist_BAG_L1 … Training model for up to 37.83s of the
57.79s of remaining time.
        Not enough time to generate out-of-fold predictions for model. Estimated
time required was 57.03s compared to 49.15s of available time.
        Time limit exceeded… Skipping KNeighborsDist_BAG_L1.
Fitting model: LightGBMXT_BAG_L1 … Training model for up to 36.69s of the
56.65s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -15.6035         = Validation score   (-mean_absolute_error)
        19.08s   = Training   runtime
        46.76s   = Validation runtime
Fitting model: LightGBM_BAG_L1 … Training model for up to 8.83s of the 28.79s
of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -17.042  = Validation score   (-mean_absolute_error)
        8.57s    = Training   runtime
        14.81s   = Validation runtime
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L2 … Training model for up to 59.86s of the
16.36s of remaining time.
        -15.5646         = Validation score   (-mean_absolute_error)
        0.1s     = Training   runtime
        0.0s     = Validation runtime
Fitting 9 L2 models …
Fitting model: LightGBMXT_BAG_L2 … Training model for up to 16.26s of the
16.25s of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -16.1537         = Validation score   (-mean_absolute_error)
        1.9s     = Training   runtime
        0.42s    = Validation runtime
Fitting model: LightGBM_BAG_L2 … Training model for up to 12.14s of the 12.13s
of remaining time.
        Fitting 8 child models (S1F1 - S1F8) | Fitting with
ParallelLocalFoldFittingStrategy
        -15.8514         = Validation score   (-mean_absolute_error)
        1.43s    = Training   runtime
        0.13s    = Validation runtime
Fitting model: RandomForestMSE_BAG_L2 … Training model for up to 8.85s of the
8.84s of remaining time.
        -15.5697         = Validation score   (-mean_absolute_error)
        17.87s   = Training   runtime
        0.55s    = Validation runtime
```

```
Completed 1/20 k-fold bagging repeats …
Fitting model: WeightedEnsemble_L3 … Training model for up to 59.86s of the
-9.78s of remaining time.
        -15.4504         = Validation score   (-mean_absolute_error)
        0.14s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 69.94s … Best model:
"WeightedEnsemble_L3"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_79_jorge_C/")
```

3 Submit

```
[ ]: import pandas as pd
import matplotlib.pyplot as plt

train_data_with_dates = TabularDataset('X_train_raw.csv')
train_data_with_dates["ds"] = pd.to_datetime(train_data_with_dates["ds"])

test_data = TabularDataset('X_test_raw.csv')
test_data["ds"] = pd.to_datetime(test_data["ds"])
#test_data
```

```
Loaded data from: X_train_raw.csv | Columns = 53 / 53 | Rows = 92951 -> 92951
Loaded data from: X_test_raw.csv | Columns = 52 / 52 | Rows = 2160 -> 2160
```

```
[ ]: test_ids = TabularDataset('test.csv')
test_ids["time"] = pd.to_datetime(test_ids["time"])
# merge test_data with test_ids
test_data_merged = pd.merge(test_data, test_ids, how="inner", right_on=["time",␣
 ↪"location"], left_on=["ds", "location"])

#test_data_merged
```

```
Loaded data from: test.csv | Columns = 4 / 4 | Rows = 2160 -> 2160
```

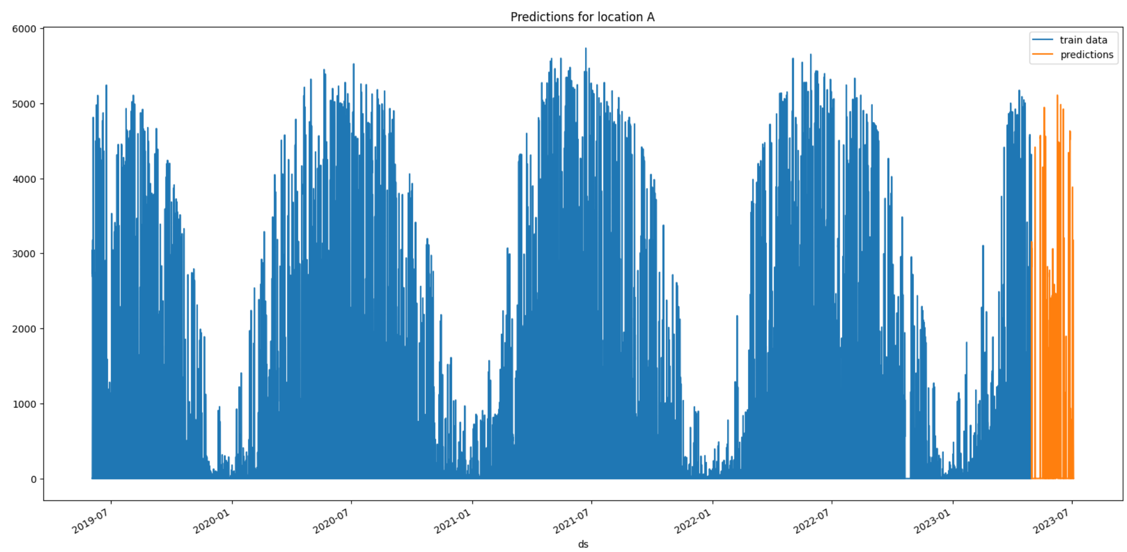
```
[ ]: # predict, grouped by location
predictions = []
location_map = {
    "A": 0,
    "B": 1,
    "C": 2
}
for loc, group in test_data.groupby('location'):
    i = location_map[loc]
    subset = test_data_merged[test_data_merged["location"] == loc].
 ↪reset_index(drop=True)
    #print(subset)
```

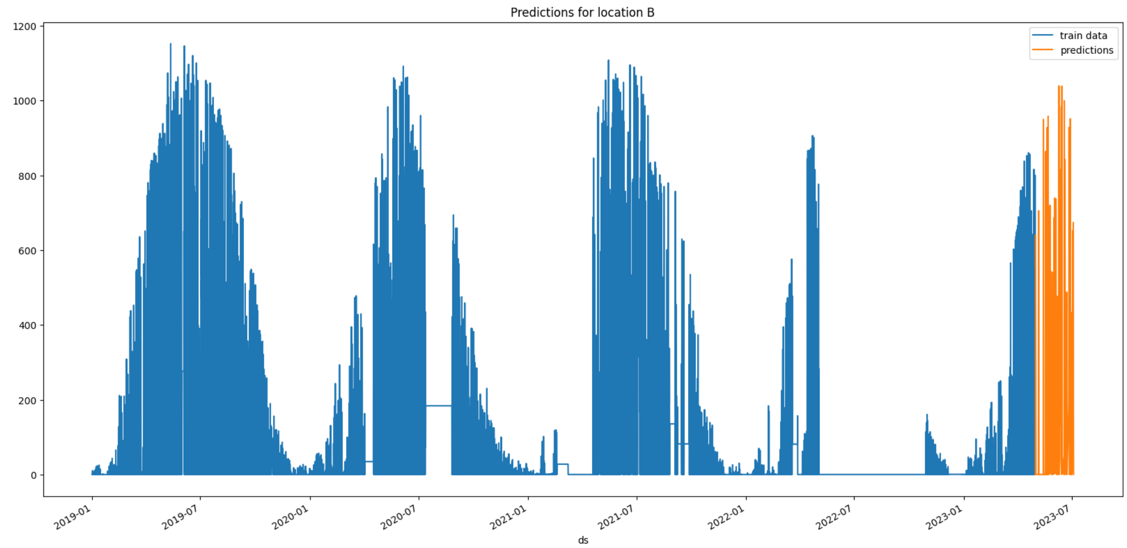
```
    pred = predictors[i].predict(subset)
    subset["prediction"] = pred
    predictions.append(subset)
```

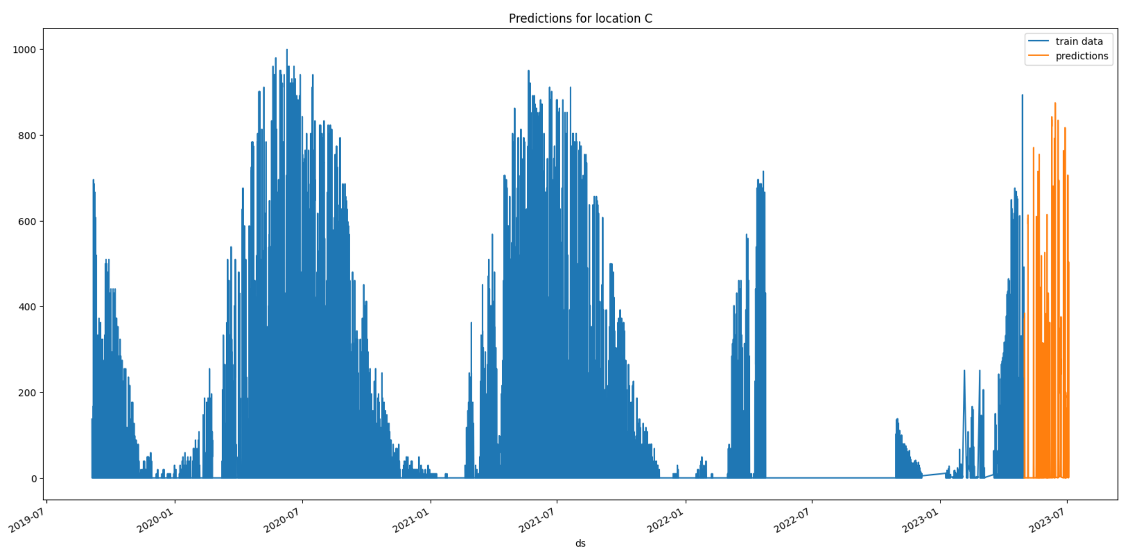
[ ]:
```
# plot predictions for location A, in addition to train data for A
for loc, idx in location_map.items():
    fig, ax = plt.subplots(figsize=(20, 10))
    # plot train data
    train_data_with_dates[train_data_with_dates["location"]==loc].plot(x='ds',␣
 ↪y='y', ax=ax, label="train data")

    # plot predictions
    predictions[idx].plot(x='ds', y='prediction', ax=ax, label="predictions")

    # title
    ax.set_title(f"Predictions for location {loc}")
```







[ ]:
```
# concatenate predictions
submissions_df = pd.concat(predictions)
submissions_df = submissions_df[["id", "prediction"]]
submissions_df
```

[ ]:

	id	prediction
0	0	1.353503
1	1	1.377337
2	2	1.535772
3	3	50.274967

```
4        4  298.462311
..     …          …
715  2155   91.790527
716  2156   58.327251
717  2157   25.278416
718  2158    3.892292
719  2159    2.036365

[2160 rows x 2 columns]
```

[ ]:
```
# Save the submission DataFrame to submissions folder, create new name based on
 ↪last submission, format is submission_<last_submission_number + 1>.csv

# Save the submission
print(f"Saving submission to submissions/{new_filename}.csv")
submissions_df.to_csv(os.path.join('submissions', f"{new_filename}.csv"),
 ↪index=False)
```

```
Saving submission to submissions/submission_79_jorge.csv
```

[ ]:
```
# save this notebook to submissions folder
import subprocess
import os
subprocess.run(["jupyter", "nbconvert", "--to", "pdf", "--output", os.path.
 ↪join('notebook_pdfs', f"{new_filename}.pdf"), "autogluon_each_location.
 ↪ipynb"])
```

```
[NbConvertApp] Converting notebook autogluon_each_location.ipynb to pdf
[NbConvertApp] Support files will be in notebook_pdfs/submission_79_jorge_files/
[NbConvertApp] Making directory
./notebook_pdfs/submission_79_jorge_files/notebook_pdfs
[NbConvertApp] Writing 152138 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 1471956 bytes to notebook_pdfs/submission_79_jorge.pdf
```

[ ]:
```
CompletedProcess(args=['jupyter', 'nbconvert', '--to', 'pdf', '--output',
'notebook_pdfs/submission_79_jorge.pdf', 'autogluon_each_location.ipynb'],
returncode=0)
```

[ ]:
```
# feature importance
location="A"
split_time = pd.Timestamp("2022-10-28 22:00:00")
estimated = train_data_with_dates[train_data_with_dates["ds"] >= split_time]
```

```
estimated = estimated[estimated["location"] == location]
predictors[0].feature_importance(feature_stage="original", data=estimated,␣
  ↪time_limit=60*10)
```

```
These features in provided data are not utilized by the predictor and will be
ignored: ['ds', 'elevation:m', 'sample_weight', 'location']
Computing feature importance via permutation shuffling for 48 features using
4394 rows with 10 shuffle sets… Time limit: 600s…
        3094.63s        = Expected runtime (309.46s per shuffle set)
        419.07s = Actual runtime (Completed 2 of 10 shuffle sets) (Early
stopping due to lack of time…)
```

[ ]:

	importance	stddev	p_value	n	p99_high \
direct_rad:W	179.527752	0.089041	0.000112	2	183.535682
clear_sky_rad:W	102.034121	0.763313	0.001684	2	136.392434
diffuse_rad:W	76.319510	0.242082	0.000714	2	87.216140
sun_elevation:d	48.306937	1.434587	0.006683	2	112.880735
hour	37.283222	0.841950	0.005082	2	75.181160
sun_azimuth:d	34.812837	0.901175	0.005826	2	75.376611
direct_rad_1h:J	28.456567	0.184929	0.001463	2	36.780617
cloud_base_agl:m	23.171833	0.354225	0.003441	2	39.116231
clear_sky_energy_1h:J	22.080754	0.614640	0.006264	2	49.747021
total_cloud_cover:p	21.249354	0.411635	0.004360	2	39.777903
effective_cloud_cover:p	17.259895	0.014190	0.000185	2	17.898613
month	16.052628	0.649876	0.009110	2	45.304928
ceiling_height_agl:m	15.125792	0.676116	0.010058	2	45.559223
diffuse_rad_1h:J	14.148167	0.011894	0.000189	2	14.683522
relative_humidity_1000hPa:p	13.232798	0.297330	0.005057	2	26.616249
is_day:idx	12.805033	0.807725	0.014188	2	49.162422
wind_speed_u_10m:ms	12.050327	0.464144	0.008667	2	32.942436
weekday	11.374554	0.931762	0.018417	2	53.315108
is_in_shadow:idx	10.136370	0.018201	0.000404	2	10.955648
msl_pressure:hPa	9.303426	0.654040	0.015810	2	38.743145
t_1000hPa:K	8.931381	0.362485	0.009132	2	25.247605
visibility:m	8.547172	0.518337	0.013641	2	31.878598
is_weekend	8.137093	0.320401	0.008860	2	22.559020
wind_speed_10m:ms	7.407126	0.125153	0.003803	2	13.040537
sfc_pressure:hPa	7.345058	0.528305	0.016175	2	31.125179
pressure_100m:hPa	6.942622	0.545800	0.017677	2	31.510253
pressure_50m:hPa	6.667661	0.192459	0.006496	2	15.330645
wind_speed_v_10m:ms	6.375395	0.245266	0.008657	2	17.415344
fresh_snow_24h:cm	5.625596	0.543399	0.021708	2	30.085121
dew_point_2m:K	4.835024	0.656765	0.030480	2	34.397422
estimated_diff_hours	4.296988	0.021430	0.001123	2	5.261618
snow_water:kgm2	4.248560	0.146461	0.007758	2	10.841074
precip_type_5min:idx	2.011457	0.735394	0.080526	2	35.113079
air_density_2m:kgm3	1.468686	0.179319	0.027413	2	9.540235

fresh_snow_12h:cm	1.399083	0.050244	0.008081	2	3.660664
absolute_humidity_2m:gm3	1.260919	0.006155	0.001099	2	1.537983
super_cooled_liquid_water:kgm2	0.855587	0.032606	0.008576	2	2.323244
snow_depth:cm	0.446138	0.040745	0.020528	2	2.280167
precip_5min:mm	0.422909	0.164053	0.085216	2	7.807265
fresh_snow_6h:cm	0.291019	0.010971	0.008483	2	0.784855
dew_or_rime:idx	0.104030	0.001518	0.003285	2	0.172374
prob_rime:p	0.038235	0.014263	0.082092	2	0.680243
snow_melt_10min:mm	0.016844	0.102404	0.427249	2	4.626258
fresh_snow_1h:cm	0.012910	0.006818	0.113768	2	0.319820
fresh_snow_3h:cm	0.002094	0.028007	0.466464	2	1.262740
wind_speed_w_1000hPa:ms	0.000000	0.000000	0.500000	2	0.000000
rain_water:kgm2	-0.032241	0.039702	0.771960	2	1.754814
year	-0.048251	0.020571	0.906799	2	0.877698

	p99_low
direct_rad:W	175.519822
clear_sky_rad:W	67.675808
diffuse_rad:W	65.422880
sun_elevation:d	-16.266861
hour	-0.614717
sun_azimuth:d	-5.750937
direct_rad_1h:J	20.132516
cloud_base_agl:m	7.227436
clear_sky_energy_1h:J	-5.585513
total_cloud_cover:p	2.720806
effective_cloud_cover:p	16.621176
month	-13.199671
ceiling_height_agl:m	-15.307639
diffuse_rad_1h:J	13.612812
relative_humidity_1000hPa:p	-0.150654
is_day:idx	-23.552357
wind_speed_u_10m:ms	-8.841782
weekday	-30.566001
is_in_shadow:idx	9.317091
msl_pressure:hPa	-20.136293
t_1000hPa:K	-7.384843
visibility:m	-14.784253
is_weekend	-6.284834
wind_speed_10m:ms	1.773715
sfc_pressure:hPa	-16.435063
pressure_100m:hPa	-17.625010
pressure_50m:hPa	-1.995323
wind_speed_v_10m:ms	-4.664554
fresh_snow_24h:cm	-18.833928
dew_point_2m:K	-24.727373
estimated_diff_hours	3.332359

```
snow_water:kgm2                    -2.343955
precip_type_5min:idx              -31.090164
air_density_2m:kgm3                -6.602862
fresh_snow_12h:cm                  -0.862498
absolute_humidity_2m:gm3            0.983856
super_cooled_liquid_water:kgm2     -0.612070
snow_depth:cm                      -1.387891
precip_5min:mm                     -6.961447
fresh_snow_6h:cm                   -0.202818
dew_or_rime:idx                     0.035686
prob_rime:p                        -0.603772
snow_melt_10min:mm                 -4.592570
fresh_snow_1h:cm                   -0.294000
fresh_snow_3h:cm                   -1.258551
wind_speed_w_1000hPa:ms             0.000000
rain_water:kgm2                    -1.819296
year                               -0.974200
```

[ ]:
```
# feature importance
observed = train_data_with_dates[train_data_with_dates["ds"] < split_time]
observed = observed[observed["location"] == location]
predictor.feature_importance(feature_stage="original", data=observed,␣
  ↪time_limit=60*10)
```

```
These features in provided data are not utilized by the predictor and will be
ignored: ['ds', 'elevation:m', 'sample_weight', 'location']
Computing feature importance via permutation shuffling for 48 features using
5000 rows with 10 shuffle sets… Time limit: 600s…
        4293.33s        = Expected runtime (429.33s per shuffle set)
        359.81s = Actual runtime (Completed 1 of 10 shuffle sets) (Early
stopping due to lack of time…)
```

[ ]:
```
                              importance  stddev  p_value  n  p99_high  \
clear_sky_rad:W                34.800616     NaN      NaN  1       NaN
sun_elevation:d                21.053896     NaN      NaN  1       NaN
clear_sky_energy_1h:J          17.959037     NaN      NaN  1       NaN
direct_rad:W                   15.484854     NaN      NaN  1       NaN
diffuse_rad:W                   8.571135     NaN      NaN  1       NaN
direct_rad_1h:J                 3.283643     NaN      NaN  1       NaN
relative_humidity_1000hPa:p     1.839748     NaN      NaN  1       NaN
sun_azimuth:d                   1.549566     NaN      NaN  1       NaN
hour                            0.997634     NaN      NaN  1       NaN
wind_speed_v_10m:ms             0.907447     NaN      NaN  1       NaN
msl_pressure:hPa                0.676269     NaN      NaN  1       NaN
snow_water:kgm2                 0.627690     NaN      NaN  1       NaN
is_day:idx                      0.620870     NaN      NaN  1       NaN
precip_type_5min:idx            0.312266     NaN      NaN  1       NaN
wind_speed_10m:ms               0.258658     NaN      NaN  1       NaN
```

pressure_100m:hPa	0.258171	NaN	NaN	1	NaN
ceiling_height_agl:m	0.255112	NaN	NaN	1	NaN
sfc_pressure:hPa	0.224268	NaN	NaN	1	NaN
pressure_50m:hPa	0.223026	NaN	NaN	1	NaN
precip_5min:mm	0.207318	NaN	NaN	1	NaN
snow_depth:cm	0.079170	NaN	NaN	1	NaN
air_density_2m:kgm3	0.078937	NaN	NaN	1	NaN
fresh_snow_12h:cm	0.077565	NaN	NaN	1	NaN
effective_cloud_cover:p	0.044103	NaN	NaN	1	NaN
snow_melt_10min:mm	0.018210	NaN	NaN	1	NaN
fresh_snow_6h:cm	0.014053	NaN	NaN	1	NaN
fresh_snow_3h:cm	0.004762	NaN	NaN	1	NaN
estimated_diff_hours	0.000000	NaN	NaN	1	NaN
wind_speed_w_1000hPa:ms	-0.000056	NaN	NaN	1	NaN
prob_rime:p	-0.000317	NaN	NaN	1	NaN
fresh_snow_1h:cm	-0.001555	NaN	NaN	1	NaN
dew_or_rime:idx	-0.003530	NaN	NaN	1	NaN
fresh_snow_24h:cm	-0.015575	NaN	NaN	1	NaN
rain_water:kgm2	-0.020746	NaN	NaN	1	NaN
year	-0.021817	NaN	NaN	1	NaN
super_cooled_liquid_water:kgm2	-0.096266	NaN	NaN	1	NaN
cloud_base_agl:m	-0.115793	NaN	NaN	1	NaN
t_1000hPa:K	-0.135829	NaN	NaN	1	NaN
wind_speed_u_10m:ms	-0.223229	NaN	NaN	1	NaN
visibility:m	-0.255399	NaN	NaN	1	NaN
is_weekend	-0.301423	NaN	NaN	1	NaN
weekday	-0.339798	NaN	NaN	1	NaN
total_cloud_cover:p	-0.380572	NaN	NaN	1	NaN
diffuse_rad_1h:J	-0.514786	NaN	NaN	1	NaN
absolute_humidity_2m:gm3	-0.560996	NaN	NaN	1	NaN
month	-0.563007	NaN	NaN	1	NaN
is_in_shadow:idx	-0.585143	NaN	NaN	1	NaN
dew_point_2m:K	-1.026145	NaN	NaN	1	NaN

	p99_low
clear_sky_rad:W	NaN
sun_elevation:d	NaN
clear_sky_energy_1h:J	NaN
direct_rad:W	NaN
diffuse_rad:W	NaN
direct_rad_1h:J	NaN
relative_humidity_1000hPa:p	NaN
sun_azimuth:d	NaN
hour	NaN
wind_speed_v_10m:ms	NaN
msl_pressure:hPa	NaN
snow_water:kgm2	NaN

```
is_day:idx                             NaN
precip_type_5min:idx                   NaN
wind_speed_10m:ms                      NaN
pressure_100m:hPa                      NaN
ceiling_height_agl:m                   NaN
sfc_pressure:hPa                       NaN
pressure_50m:hPa                       NaN
precip_5min:mm                         NaN
snow_depth:cm                          NaN
air_density_2m:kgm3                    NaN
fresh_snow_12h:cm                      NaN
effective_cloud_cover:p                NaN
snow_melt_10min:mm                     NaN
fresh_snow_6h:cm                       NaN
fresh_snow_3h:cm                       NaN
estimated_diff_hours                   NaN
wind_speed_w_1000hPa:ms                NaN
prob_rime:p                            NaN
fresh_snow_1h:cm                       NaN
dew_or_rime:idx                        NaN
fresh_snow_24h:cm                      NaN
rain_water:kgm2                        NaN
year                                   NaN
super_cooled_liquid_water:kgm2         NaN
cloud_base_agl:m                       NaN
t_1000hPa:K                            NaN
wind_speed_u_10m:ms                    NaN
visibility:m                           NaN
is_weekend                             NaN
weekday                                NaN
total_cloud_cover:p                    NaN
diffuse_rad_1h:J                       NaN
absolute_humidity_2m:gm3               NaN
month                                  NaN
is_in_shadow:idx                       NaN
dew_point_2m:K                         NaN
```

[ ]:
```
subprocess.run(["jupyter", "nbconvert", "--to", "pdf", "--output", os.path.
↪join('notebook_pdfs', f"{new_filename}_with_feature_importance.pdf"),␣
↪"autogluon_each_location.ipynb"])
```

```
[NbConvertApp] Converting notebook autogluon_each_location.ipynb to pdf
[NbConvertApp] Support files will be in
notebook_pdfs/submission_79_jorge_with_feature_importance_files/
[NbConvertApp] Making directory
./notebook_pdfs/submission_79_jorge_with_feature_importance_files/notebook_pdfs
[NbConvertApp] Writing 152954 bytes to notebook.tex
[NbConvertApp] Building PDF
```

```
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 1471953 bytes to
notebook_pdfs/submission_79_jorge_with_feature_importance.pdf
```

[ ]:
```
CompletedProcess(args=['jupyter', 'nbconvert', '--to', 'pdf', '--output',
'notebook_pdfs/submission_79_jorge_with_feature_importance.pdf',
'autogluon_each_location.ipynb'], returncode=0)
```

[ ]:
```
import subprocess

def execute_git_command(directory, command):
    """Execute a Git command in the specified directory."""
    try:
        result = subprocess.check_output(['git', '-C', directory] + command,␣
 ↪stderr=subprocess.STDOUT)
        return result.decode('utf-8').strip(), True
    except subprocess.CalledProcessError as e:
        print(f"Git command failed with message: {e.output.decode('utf-8').
 ↪strip()}")
        return e.output.decode('utf-8').strip(), False

git_repo_path = "."

execute_git_command(git_repo_path, ['config', 'user.email', 'henrikskog01@gmail.
 ↪com'])
execute_git_command(git_repo_path, ['config', 'user.name', hello if hello is␣
 ↪not None else 'Henrik eller Jørgen'])

branch_name = new_filename

# add datetime to branch name
branch_name += f"_{pd.Timestamp.now().strftime('%Y-%m-%d_%H-%M-%S')}"

commit_msg = "run result"

execute_git_command(git_repo_path, ['checkout', '-b',branch_name])

# Navigate to your repo and commit changes
execute_git_command(git_repo_path, ['add', '.'])
execute_git_command(git_repo_path, ['commit', '-m',commit_msg])

# Push to remote
```

```
output, success = execute_git_command(git_repo_path, ['push',␣
↪'origin',branch_name])

# If the push fails, try setting an upstream branch and push again
if not success and 'upstream' in output:
    print("Attempting to set upstream and push again...")
    execute_git_command(git_repo_path, ['push', '--set-upstream',␣
↪'origin',branch_name])
    execute_git_command(git_repo_path, ['push', 'origin', 'henrik_branch'])

execute_git_command(git_repo_path, ['checkout', 'main'])
```

[ ]:
```
('Switched to branch \'main\'\nYour branch is behind \'origin/main\' by 2
commits, and can be fast-forwarded.\n  (use "git pull" to update your local
branch)',
 True)
```